We will measure the effectiveness of K-NN by testing the trained model with new data, steps are as follows:

1. Split the dataset(n) into 2 dataset training dataset(n1) and test dataset(n2).

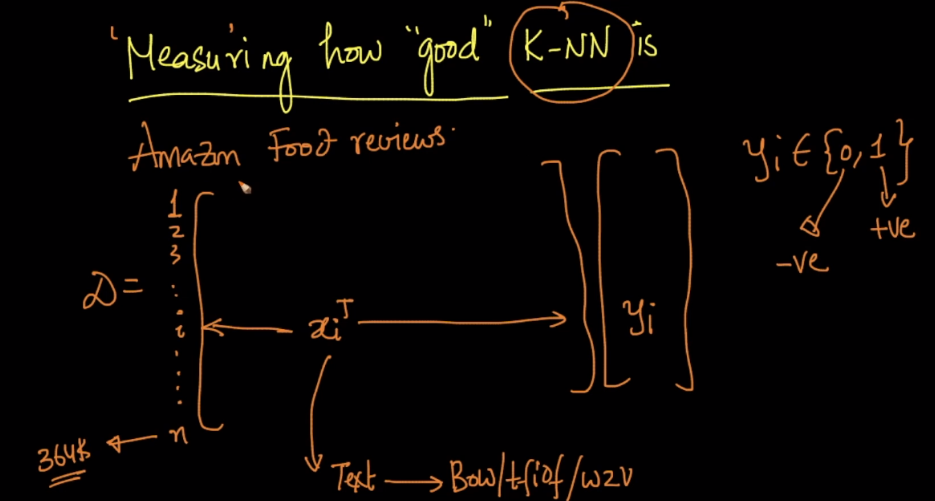
n1 will be 70% of n

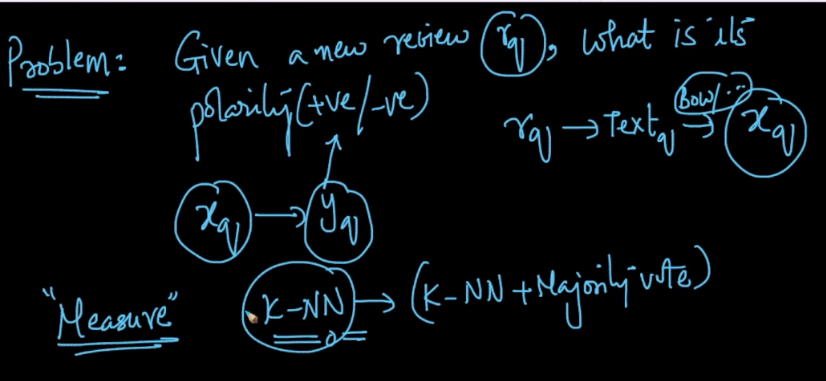
n2 will be 30% of n

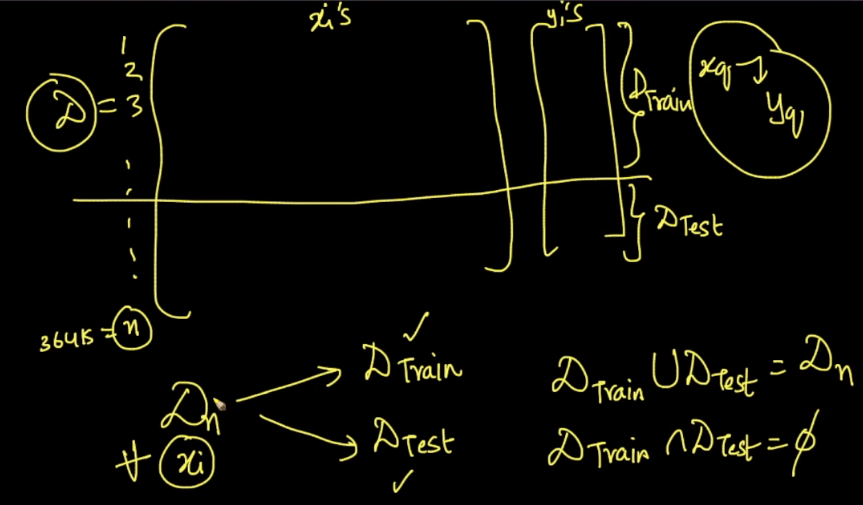
We split data among training and test dataset randomly.

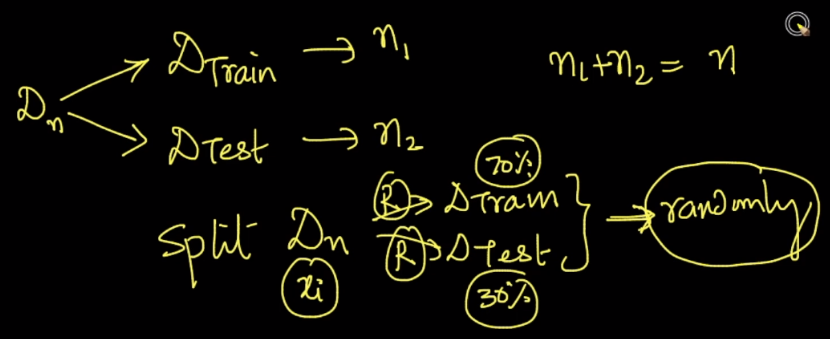
1. Apply k-NN on training dataset to train k-NN model, so now our K-NN model knows all the clusters of points(according to training dataset)
2. Following operations are performed:

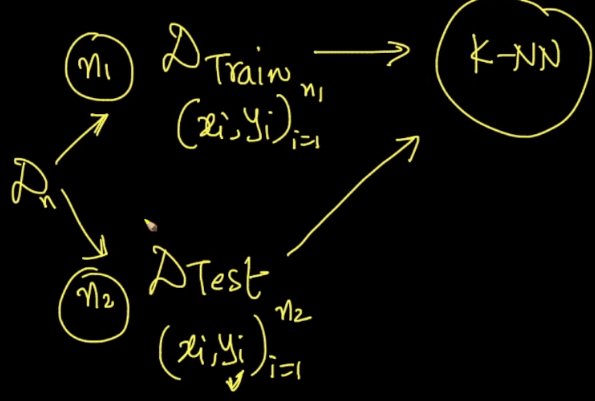
* Our test dataset have similar features as of training dataset along with it’s original class/output(Ypt).
* Now we apply trained k-NN on each datapoint of test dataset, this will gives an output class(Yq)
* Now for all we count how many data points in test dataset get output from k-NN similar to it’s original class.
* Now we diveide the count we obtain in above step by total no of datapoints in test dataset, which will give us the percentage of accuracy of our k-NN model











**Notes:**

* What should be the value for ‘k’ in K-NN.

There is no correct range of 'K' that is valid for all the problems. It depends on the problem we are solving and the size of the dataset and number of dimensions. You have to keep performing hyper-parameter tuning for a wider range of values (say from 1 to 50 (or) 1 to 100) and find the optimal value that prevents model from both overfitting and underfitting.

When Train Error is high and CV Error is high --> Model Underfits

When Train Error is low and CV Error is high --> Model Overfits

